# Towards a Human-AI Hybrid System for Categorising Programming Problems

Filipe Dwan Pereira
Francisco Pires
Federal University of Roraima
Boa Vista, Brazil
filipe.dwan@ufrr.br,junior-pires.rr@hotmail.com

Samuel C. Fonseca
Elaine H. T. Oliveira
Leandro S. G. Carvalho
David B. F. Oliveira
Federal University of Amazonas
Manaus, Brazil
{scf,elaine,galvao,david}@icomp.ufam.edu.br

Alexandra I. Cristea
Durham University
Durham, UK
alexandra.i.cristea@durham.ac.uk

## Abstract

As programming skills are increasingly required world-wide and across disciplines, many students use online platforms that provide automatic feedback through a Programming Online Judge (POJ) mechanism. POJs are very popular e-learning tools, boasting large collections of programming problems. Despite their many benefits, students often struggle when solving problems not compatible with their prior knowledge. One important cause of this is that usually statements of problems are not classified according to programming topics (paradigms, data structures, etc.) and, hence, students waste time and effort in trying to solve exercises that are not tailored to their level and needs. Thus, to support students, we propose a new, "front-heavy" pipeline method to predict topics of POJ problems, using Bidirectional Encoder Representations from Transformers (BERT) for contextual text augmentation over the problem statements and further allowing for (lighter-weight) classical machine learning for classification. Our model outperformed all current state-of-the art, with an F1-score of ≈ 86% using stratified 10 fold cross-validation in a classically challenging multi-classification problem with seven categories. As a proof of concept, we conducted an experiment to show how our predictive model can be used as a human-AI hybrid complement for POJ, where learners would use AI-based recommendations to find the most appropriate problems.

## CCS Concepts

• **Applied computing** → **Computer-assisted instruction**; *Annotation*; • **Computing methodologies** → *Natural language generation*; *Supervised learning by classification.*

**ACM Reference Format:**
Filipe Dwan Pereira, Francisco Pires, Samuel C. Fonseca, Elaine H. T. Oliveira, Leandro S. G. Carvalho, David B. F. Oliveira, and Alexandra I. Cristea. 2021. Towards a Human-AI Hybrid System for Categorising Programming Problems. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432422


*SIGCSE '21, March 13–20, 2021, Virtual Event, USA*

ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432422

## 1 Introduction

Programming Online Judge (POJ) provides a reliable automatic and instantaneous evaluation of the source code of an algorithm sent by the learners [40]. POJs are platforms used by many learners who wish to improve their programming skills. These systems have been shown to enhance education, promote competitive programming and support recruitment processes [5, 6, 35, 39, 41]. They are widely used in job interviews of large technology companies, such as Google, Amazon, etc. [40, 41]. Originally, POJs were designed for self-directed learning, without the aid of an instructor [40, 41]. However, these systems are increasingly being used in education institutions (e.g. to support programming classes) as they reduce instructors' workload in correcting the learners' programming tasks. At the same time that provides students with instantaneous feedback about the correctness of their solutions [35, 40].

POJ problems span over different computer science categories – e.g., graphs, paradigms, computational geometry, data structures, etc. Understanding such categorisation allows learners to engage with specific algorithm techniques and could improve their skills. Furthermore, for curriculum design, it has been previously pointed out that it is essential to consider problem categories [38, 41].

Nonetheless, this massive variety of problems are typically piled in volumes which are not organised by categories or difficulty levels [5, 17, 39, 41]. Zhao et al. [41] explain that organising problems by volume has low maintenance costs for developing and updating a POJ system, since many problems could be easily collected (e.g., from an annual ACM International Collegiate Programming Contest (ICPC)) and piled into a new volume, without any classification. However, such convenience for POJ maintainers comes with an educational cost for learners, as they may loose time and waste effort in trying to solve exercises that are not tailored to their level and needs, what might lead to frustration and dropout. To illustrate, it is hard and sometimes frustrating for learners to find problems of, e.g., computational geometry, which may be spread across multiple volumes. Similarly, novice programmers tend to struggle to find easy problems in their favourite categories.

One reason why these problems are not categorised is that this task of annotation is manually performed by the problems' creators or expert users, which is a costly and tedious process [6, 12, 41]. In addition, this process is not scalable, as POJs are continuously increasing their collection of problems [17, 40]. Thus, we formulate the following research question: *how to support semi-automatic categorisation of programming problems from online judges?*

To answer our research question, we trained and compared different predictive models, capable of detecting the categories of the problem based on their statements through the use of advanced Natural Language Processing techniques (NLP) and Machine Learning (ML). Moreover, we propose a method able to use small data, by using contextual paraphrasing with BERT, for text augmentation on our training set, to boost the performance.

Furthermore, using our predictive model, we conducted an experiment with learners to verify to which extent our model can be useful in helping learners to recognise the problems' category and, hence, the programming topic needed to solve a problem. As a result, our experiments showed that our model could support POJ users in the task of categorisation and, hence, the model can be used as a human-AI hybrid complement for POJ systems, to arrange and categorise problems, where learners would use AI-based recommendations to find the most appropriate exercises. To sum up, our main contributions are:

- a new 'front-heavy' pipeline method to predict topics of POJ problems with BERT for contextual text augmentation for small data;
- shallow learning for the main multiple-class classification problem, with higher performance than SoA.
- empirical evidence that our method can be used as a human-AI hybrid complement for POJ;
- a new *dataset*[0] collected and refined from 2 POJ that can be used as benchmark in future works.

## 2 Related Work

Text mining in educational scenarios is gaining momentum, due to the increasing availability of different sources, such as social media, discussion forums in MOOC systems, online news, etc. [2, 10, 12]. In spite of this boom, surprisingly few works analysed problem descriptions in POJs. Instead, POJ studies use student data – e.g., the number of accepted problems, wrong answers, or compilation errors – to propose methods to help in learning programming. Such data is used to predict learner performance [1, 3, 22, 31, 32, 34, 36], estimate dropout [21, 33], recommend tasks [5, 18, 39], or detect problem difficulty [9, 17, 24]. Next, we look at the (very small subset of) POJ studies targeting the same problem as us – that of predicting problem categories (also called problem topics) in online judges.

A recent pioneering study tackling this task [41] used students' sequence of attempts to solve programming problems, to predict the categories of 940 problems collected from three POJs. The authors depicted the multi-classification task with 10 categories; their best model achieved an overall F1-score of ≈ 70.63%, which we believe may be due to their selected categories not being appropriate (e.g., their 'Tricky problem' category would clearly overlap with several others, etc.). Further, [41] also tried to use problem statements to extract categories; however, they achieved poor results (≈ 40% accuracy) using Latent Dirichlet Allocation (LDA). In spite of the result, [41] recognised the potential of using text mining on problem descriptions, but argued that it might be hard to extract useful information from these problems statements, as they might contain figures or attachments. To check their argument, we randomly selected 100 problems from the A2 online judge and only 13 problems had figures. Moreover, even removing those figures, it is possible to understand the problem and categorise it. Athavale et al. [4] also addressed the problem of category prediction via natural language techniques over problem statements. Their study used two datasets for the multi-class classification, with a total of 1709 problems distributed onto four categories. However, their best predictive models achieved only F1-scores between 19.2% to 62.2%, 9% lower than human-level analysis.

Instead, here we propose a new pipeline model, which combines cutting edge NLP techniques, such as BERT, with different shallow and deep ML classifiers. We carefully design a front-heavy method, to deal with sparse categories, in the text preprocessing – here, augmentation phase – but allow for further training to be done on lightweight systems, if new data is collected. We apply our new model on the challenging multi-class problem with a total of 7 categories, onto two new datasets, and show that we achieve a higher performance, when compared to the state of the art (SoA) [4, 41]. Moreover, we go one step further and, for the first time, to the best of our knowledge, show via experiments with humans that our model could support learners in problem categorisation.

## 3 Research Design

### 3.1 Problem Definition

POJs automatically evaluate code submitted by learners as solutions to programming problems [40]. The evaluation process uses test cases to check whether the output of the learner code matches the expected outputs [33, 40]. Statements of programming problems in POJ have a description (or various length) and some example Test Cases. Figure 1 illustrates an example problem. We chose this (easy) problem as it is a rare case of a problem with a brief statement.

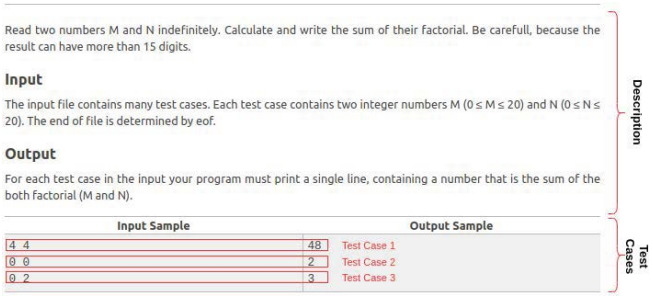


**Figure 1: A problem statement from the URI Online Judge**

In this work, we are interested in grouping problems in terms of their related programming knowledge components, concepts or skills. To illustrate, a problem that can be solved by using a stack or a linked list, can be classified as belonging to the 'data structure' category. Similarly, a problem that employs dynamic programming or backtracking belongs to the class paradigms. A very popular book among ICPC competitors [16] defines 7 macro-categories of programming problems, which are: introductory problems, data structure, paradigms, graphs, mathematics, strings processing, computational geometry.

Thus, we model this task of categorisation as a multi-classification problem, i.e, our predictive model must estimate one out of $n$ categories for each problem statement (in our case, $n$=7, the seven categories aforementioned). Our predictive model can be represented by the function $f : P \rightarrow C$, where $P$ denote the set of problems statements and $C$ the categories $C = \{c_1, c_2, ..., c_7\}$, with:

[0] github.com/filipedwan/SIGCSE2021-nlp-poj

**Input:** a problem statement $p \in P$.
**Output:** a predicted category $c' \in C$, where $f(p) = c'$.
**Target:** a target (correct) class $c \in C$; if $c = c'$, our prediction is correct.

## 3.2 Categorisation of Problems

As one of the contributions of this work, we scraped 5213 programming problem statements, merged and unified the labels from the URI Online Judge (URI for short) and A2 Online Judge (A2 for short). In both systems, the problem categories were annotated either by the problem creators, or by expert users, who had already solved the problem.

Some issues occurred from the start, as the problems were categorised differently in the two systems. Whilst URI used 8 computer science (macro-)subjects, A2 used finer-grained segregation, with a total of 152 categories. The reason for so many categories was that the maintainers manually labelled the problems with a plethora of specific categories, such as minimum spanning tree, depth-first search, flood fill, which could easily be classified into a more general category, such as graph. Such extremely detailed categorisation led also to extreme sparsity, with some classes with 1-2 problem instances only. To be more precise, 75 of the A2 categories contain less than 10 problems, rendering the data hard for machine learning. Note that with exception of the ad-hoc category (explained next), a well-known book [16] only recommends a limited number of categories – the ones used by URI as macro-categories. Thus, we map the categories of A2 over the macro-categories of URI. Table 1 presents the categories' names, categories' descriptions and the number of problems collected in each category after the mapping.

**Table 1: Database description of all problems from URI and A2 after merger**

| Category | Description | N |
|---|---|---|
| Beginner | easy problems targeted for novice programmers | 305 |
| Ad-hoc | problems involving simulation, dates, logic, and so forth | 603 |
| Strings | string manipulation, palindromes, and longest common subsequence | 387 |
| Data Structures (DS) | queue, stack map, set, hash tables, priority queues and so forth | 647 |
| Mathematics | number theory, prime numbers, combinatorics, big numbers and so forth | 1075 |
| Paradigms | dynamic programming, binary search, greedy, backtracking and so forth | 1441 |
| Graph | flood fill, minimum spanning tree, maximum flow, trees | 363 |
| Computational Geometry (CG) | problems about points, lines, polygons, etc. | 392 |

After analysing the problems from each category, we observed that the 'ad-hoc' category could be confusing. As shown in Table 1, this category comprises problems of logic, simulation, dates, which might be present in other categories. Indeed, just like a human, a machine learning model might be addled to classify problems in it, as almost all problems have some ad-hoc properties (e.g., logic) which might lead to problems with more than one category. As we are designing our problem as a multi-classification task (as URI and A2 do), not as a multi-labelling task, we thus decided to suppress the ad-hoc category from our analysis, to avoid misclassification and confusion in our test with humans. Moreover, the ad-hoc category is not present in the reputed POJ-related book [16].

## 3.3 Dealing with Class Imbalance

One known issue in deep (or shallow) learning is class imbalance, i.e., it is critical to have a sufficient number of instances for each target category. Otherwise, the estimation model for category importance may be biased. This sparsity issue can be more of a problem in multi-category classification problems such as ours. In these situations, studies [7, 15] suggest adjusting the model by changing the performance metric (e.g. using F1-score) and by trying to balance the data using undersampling or oversampling techniques.

Undersampling is not feasible in our case, as this technique is suitable only for problems with an abundance of data [26] and, thus, we opted for oversampling. In general, studies simply repeat instances of minority classes to carry out oversampling [15, 26]. However, such simple duplication of data might lead to overfitting [7]. As such, differently, we applied contextual text augmentation over the problem statements of the minority categories of our dataset, to produce new programming problem statements on the training set. To do so, we used a word embeddings augmenter [25], which stochastically replaced a token (in our case a word), by using the surrounding tokens, through the use of the cutting-edge method BERT [8]. It is worth noting that for the problem we are tackling, paraphrasing might help our ML models to analyse different forms of the same programming statement, increasing the variance of the training set. To illustrate, we show an example of a part of an original problem statement from the *beginner* category, followed by its contextual paraphrasing:

- Original sentence: *Read two integer values. After this, calculate the product between them and store the result in a variable named PROD...*
- Contextual paraphrase: *Add an arbitrary number of integer values. Given that, calculate the result with them; store the result in a variable labelled prod...*

Note that, in some cases, the contextual paraphrase does not make much sense. However, our goal here is to create more instances to increase variation on the minority classes and reduce the chances of the model to minimise errors by frequently predicting the majority class – achieving a high (but misleading) overall accuracy, but with a low recall and precision per class, mainly in the minority classes.

We were careful however with the paraphrasing technique to at most duplicate the number of samples of the minority categories on the training set, to avoid adding too much artificial information, rendering it non-representative. This approach was validated by us empirically. Thus, we decreased the imbalanced proportion, but did not completely balance the data. I.e., given our majority class *Paradigms* (1441 problems), we duplicated the number of samples in our minority class, *Beginner* (305 problems) to 610 problems. However, for *Mathematics* (1075 problems), we just created enough problems to achieve the same number as the majority class.

## 3.4 Preprocessing and Feature Extraction

Several aspects related to data quality can influence the performance of the machine learning systems. As such, in this step, we aimed to clean up the text and prepare it for the machine learning

algorithms. First, to expedite the comparisons in the indexing process, we converted each capital letter of the data to lowercase. The next step was to carry out the tokenisation of the text, meaning here breaking the sequence of words into (fragments of) single words. Subsequently, we removed the stopwords, masked the numbers, and replaced line breaks with simple spaces. Moreover, we removed all the html tags from the text.

After preprocessing, we extracted features from the text by using two techniques: i) distributed word representation and ii) analysis of word importance. For the first, we employed the famous Skip-Gram model [28] on our own corpus, to produce a word embedding for each word of our vocabulary through the use of shallow neural networks. To do so, we used the word2vec (W2V) module from the well-known gensim library with a default dimensionality of the word vectors of 20; and a default maximum distance between the current and predicted word within a sentence of 10. Still, as an alternative for distributed word representation, we extracted the word embeddings of the pre-trained GloVe [30], coined from Global Vectors, corresponding to the vocabulary of our corpus. We also used analysis of word importance, employing TF-IDF. All of these word representations were used in the machine learning algorithms to compete with each other.

### 3.5 Classification and Validation

Following previous research [11, 19, 27] in text classification, we evaluated the ensemble methods *Random Forest* (RF), Extra Trees Classifier (ETC), and XGBoost (XGB). Finally, besides using BERT for text augmentation, as explained in section 3.3, we also analysed BERT for classification (called Bert Classifier), using 12 encoding layers from a Transformer network, each layer having 12 attention heads, as recommended in [8].

For validation of the machine learning models, we used *StratifiedKFold* (from scikit-learn) with 10 folds. This method divides the data into homogeneous subgroups called stratum (stratified sampling), so that the right number of instances is sampled from each stratum, to keep the same category proportion in the training and validation sets [13]. This method is recommended [13] for imbalanced datasets like ours. For each stratified fold, we applied contextual paraphrasing only on the training set.

### 3.6 Evaluation by Humans

Predictive models can estimate the probability that an item belongs to a particular class. As such, to validate whether our method can support learners, we conducted an experiment with 10 Computer Science undergraduate students with experience in algorithms, who have already participated in ICPC. More specifically, these students participated in the regional contest of ICPC'17 (see icpc.baylor.edu/community/history). We chose these learners because they are familiar with all categories. As a controlled treatment, we asked each learner to categorise a sample of 10 programming problems manually. In an experimental treatment, the same students categorised the same problems using the predicted probabilities of our model as a heuristic. Notice that we sampled the problems (stratified sampling) from our validation set for this test with humans. Thus, the heuristic is estimated and calculated on test data.

## 4 Results and Discussion

The results of the ensemble methods RF and ETC using different text representation (W2C, GloVe, and TF-IDF) were similar, with an advantage for RF. On the other hand, XGBoost achieved lower performances. Figure 2 shows the average of the weighted F1-scores of the models on the stratified cross-validation with 10 folds.

The contextual paraphrasing boosts the performance of the predictive models in almost all cases. Notice that BERT was crucial for this boosting through the text augmentation; however, for classification, this deep learning model did not achieve the best result (see BERT and BERT+PAR in Figure 2). Indeed, Random Forest using Word2Vec and contextual paraphrases (RF+W2C+PAR) outperform the other models with the highest F1-score ($\approx$ 86%) and a low standard deviation ($p-value < 0.05$ – McNemar's Test). Still, this model surpassed the current state of the art (as explained in related works) on similar data. This represents thus one of the contributions of our work - a new, "front-heavy" pipeline method to predict topics of POJ problems with BERT for contextual text augmentation.

Figure 3 (left) shows the confusion matrix of our best model, in which rows represent true categories, whilst columns represent predicted categories. From a visual inspection, we can see that the model can achieve high performance ($> 75\%$) for the categories *computational geometry, data structure, mathematics, paradigms*, and *strings*. On the other hand, the performance for the *graph* and *beginner* categories could be better.

To further analyse the types of errors this model makes, we compare these errors proportionally. In Figure 3 (right) we fill the main diagonal with zeros to focus on the misclassification and divide the number of errors on the rows by the number of instances for each corresponding category. First, some rows are brighter, such as the rows corresponding to categories strings, DS, CG, mathematics and paradigms. This means that most of the instances from these categories are classified correctly. Notice that the errors are not perfectly symmetrical; for example, there are more *beginner* problems misclassified as *mathematics* than the reverse. Hence, columns for the categories *beginner, mathematics* and *paradigms* are darker, which tells us that many problem statements are misclassified into these three categories.

A possible explanation of why many instances were wrongly classified as *mathematics* is that some problems use some mathematical formulas, even when they do not belong to the mathematics category, which might cause misclassification. Specifically, we can see in this figure that the model confuses *beginner* and *mathematics*. Indeed, many beginner problems have as context basic mathematics operations, such as asking students to multiply variables or problems evolving factorials, Fibonacci series or counting of prime values. We can also see some confusion between *graph* and *paradigms*, which could be explained as there are graph problems which might be solved using techniques from paradigms, such as dynamic programming (e.g., the Bellman-Ford algorithm for the shortest path problem with negative edges in a graph).

It is worth noting that, in general, our model performs better on the categories which comprise the most difficult problems. To illustrate, our model achieves a recall of 89% on *computation geometry*, which contains problems evolving Convex Hull, Graham Scan, etc. Similarly, our model achieves also 89% on the *paradigms* category, which comprises problems evolving dynamic programming and backtracking, which are notorious difficult to all but experienced programmers.

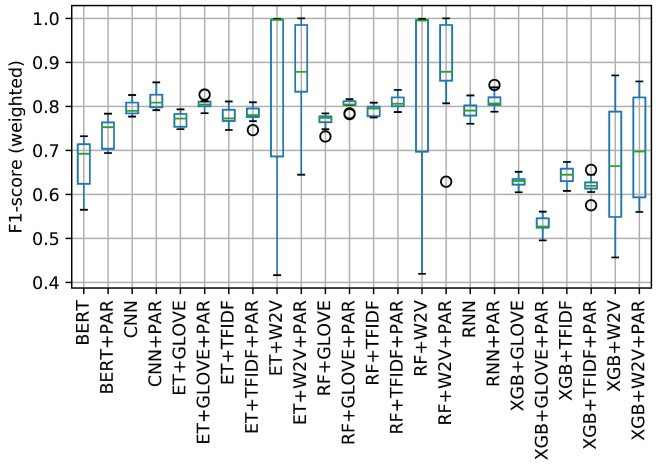


**Figure 2: We plot the weighted average F1-score since our dataset is imbalanced and this metric considers the proportion of each class for calculation. In this figure, PAR refers to the use of paraphrasing in the training set.**

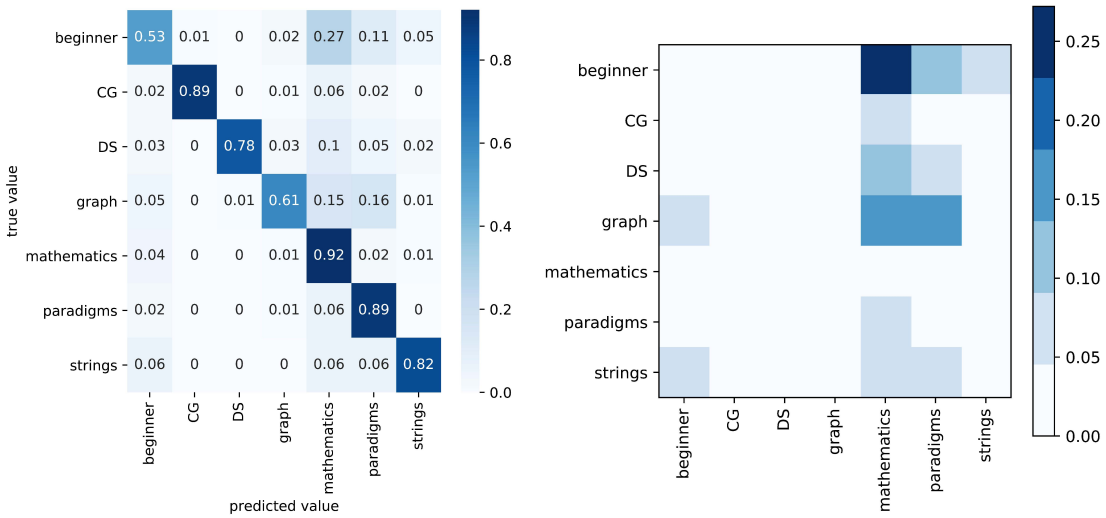


**Figure 3: Confusion Matrix of our best model (left) and error density by category (right).**

From this analysis, we hypothesise that our model can be complementary to humans in a human-AI system, as humans are better in recognising the beginner category, where the model performed worse, whilst learners with less experience would tend to struggle to classify problems of the other categories, where our model performed better.

It is important to note that Random Forest, with the best result, can estimate the probability that an item belongs to a particular class by averaging the probabilities estimation of its constituent decision trees. Thus, we checked whether these predicted probabilities can be used as a heuristic to enhance the learners capabilities' to categorise programming problems and, hence, better recognise the prior knowledge and skills needed to solve a programming task.

In the controlled treatment, without our heuristic, the learners achieved an average F1-score of ≈ 54%. Figure 4 (left) shows their performance by category in a confusion matrix. In the experimental treatment, the same students categorised the same problems using the probability estimation of our best model as a heuristic. We presented the probability estimation as an array, containing a row per problem statement and a column per category, each containing the probability that the given problem statement belongs to the given category (e.g., 70% chance that problem A belongs to the category 'graphs'). This time, the students achieved an average F1-score of ≈ 87%. Figure 4 (right) shows the confusion matrix of the students with our heuristic.

From the learner performance, we find further support towards our hypothesis that our model can be complementary for humans in POJ problem categorisation – another contribution of our work. Figure 4 shows that the learners boost their performance in all classes. Moreover, for 'tougher' categories, such as paradigms, their recall tripled. We believe that this is an important move toward the construction of a human-AI hybrid POJ system, where learners would use AI-based recommendations to find problems that are related to their skills and prior knowledge. In addition, we believe

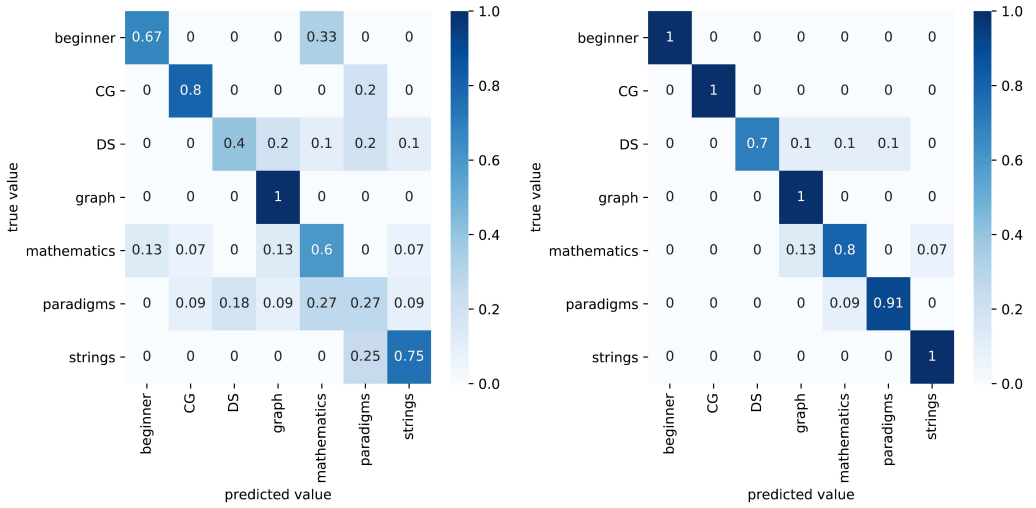


**Figure 4: Performance of competitors without (left) and with the heuristic (right).**

that our method can be applied in other domains, that is, other kinds of online systems that often have the same issues as the POJ problems: they are not (or not enough) labelled with the topic of the resource. These online systems might also benefit from our front-heavy pipeline method for categorisation.

## 5 Pedagogical Implications

Programming skills are really important nowadays and many companies are demanding qualified programmers [23, 40]. Moreover, we learn programming by doing and, hence, practicing is one of the most effective ways of improving [37]. POJ systems allow practicing with instantaneous feedback; however, a major issue of these systems is the typical lack of problem categorisation [17, 41].

For *self-direct learning*, understanding the problem categories allows learners to engage with specific algorithm techniques and could help improve their skills at a faster rate. On the other hand, without awareness of the problem category, learners might waste precious time and effort in trying to solve exercises that are not tailored to their level and needs. As an analogy, it is similar to a virtual store, where there are no salespeople physically present, and where products are not categorised, that is, there is no cue to guide customers to find out what they need, who likely will buy at another store. In education, such lack of organisation might further lead to frustration or even to dropout.

Thus, our front-heavy pipeline method and approach of using its predicted probabilities as heuristic has potential benefits for education, by enabling students to select problems that are better suited to their skill level and prior knowledge. Instructors can also benefit from that, by selecting adequate problems to compose assignment lists – in which it is essential to consider problem categories.

Finally, categories are a form of simplified ontology [14]. Ontological mapping of knowledge can lead to the understanding of the meta-knowledge ('knowledge about knowledge'), and thus allow access to a much deeper comprehension. Students accessing such knowledge know in fact not only the basic knowledge [20, 29], but also what they know – and are arguably better equipped to further extend their comprehension in an orderly and systematic fashion. Future research could further look into generating more hierarchical ontologies with increased structure and depth.

## 6 Conclusions, Limitations and Future Work

In this work, we have proposed and evaluated a new, "front-heavy" pipeline method to predict topics of POJ problems for "small (sparse) data", allowing for shallow learning for the main multi-class classification task. We showed here that our method can support online judge users in categorisation, through a heuristic, based on the probability estimation of our model. We believe that this is an important move toward the construction of a human-AI hybrid POJ system, where learners could have a better user experience in finding appropriate problems. Interestingly, our model achieved higher performance in the categories where humans achieve lower performance and vice-versa. This renders our model naturally as a useful tool for "pre-categorisation", which could be later validated with a human pipeline of expert users. As a replacement of the current manual categorisation by expert users, our model clearly allows for the task of categorisation to be accomplished with considerably less effort.

Moreover, our method can also be useful as a benchmark (as we make available our dataset) for future work. We achieve a competitive F1-score of $\approx 86\%$, above the current state of the art. Still, despite our model having proven to be a good heuristic, it can be improved upon. Further work can look into optimising techniques towards complete automatisation.

The main limitation of this work is related to the dataset. There are very few POJs that categorise their problems and, hence, it is difficult to find fully labelled datasets. Furthermore, this task of categorisation could be further extended to a multi-label classification, where one problem could potentially be labelled with multiple labels/classes. Our paraphrasing technique sometimes produced semantically irrelevant sentences. Nevertheless, this method of data balancing produced better results than simple synonym replacement, for instance. Future work can look into other methods with increased semantics.

## Acknowledgements

This research, carried out within the scope of the Samsung-UFAM Project for Education and Research (SUPER), according to Article 48 of Decree nº 6.008/2006 (SUFRAMA), was partially funded by Samsung Electronics of Amazonia Ltda., under the terms of Federal Law nº 8.387/1991, through agreements 001/2020 and 003/2019, signed with Federal University of Amazonas and FAEPI, Brazil. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

## References

[1] Alireza Ahadi, Raymond Lister, Shahil Lal, Juho Leinonen, and Arto Hellas. 2017. Performance and consistency in learning to program. In *Proceedings of the Nineteenth Australasian Computing Education Conference*. 11–16.

[2] Tahani Aljohani, Filipe Dwan Pereira, Alexandra I Cristea, and Elaine Oliveira. 2020. Prediction of Users' Professional Profile in MOOCs Only by Utilising Learners' Written Texts. In *International Conference on Intelligent Tutoring Systems*. Springer, 163–173.

[3] Michael Mogessie Ashenafi, Giuseppe Riccardi, and Marco Ronchetti. 2015. Predicting students' final exam scores from their course activities. In *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[4] Vinayak Athavale, Aayush Naik, Rajas Vanjape, and Manish Shrivastava. 2019. Predicting Algorithm Classes for Programming Word Problems. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*. Association for Computational Linguistics, Hong Kong, China, 84–93.

[5] Giorgio Audrito, Tania Di Mascio, Paolo Fantozzi, Luigi Laura, Gemma Martini, Umberto Nanni, and Marco Temperini. 2019. Recommending Tasks in Online Judges. In *International Conference in Methodologies and intelligent Systems for Techhnology Enhanced Learning*. Springer, 129–136.

[6] Jean Luca Bez, Neilor A Tonin, and Paulo R Rodegheri. 2014. URI Online Judge Academic: A tool for algorithms and programming classes. In *2014 9th International Conference on Computer Science & Education*. IEEE, 149–152.

[7] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106 (2018), 249–259.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Tomáš Effenberger, Jaroslav Čechák, and Radek Pelánek. 2019. Measuring Difficulty of Introductory Programming Tasks. In *Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale*. 1–4.

[10] Rafael Ferreira-Mello, Máverick André, Anderson Pinheiro, Evandro Costa, and Cristobal Romero. 2019. Text mining in education. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9, 6 (2019), e1332.

[11] Irina Fishcheva and Evgeny Kotelnikov. 2019. Cross-Lingual Argumentation Mining for Russian Texts. In *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 134–144.

[12] Samuel C Fonseca, Filipe Dwan Pereira, Elaine HT Oliveira, David BF Oliveira, Leandro SG Carvalho, and Alexandra I Cristea. 2020. Automatic Subject-based Contextualisation of Programming Assignment Lists. Educational Data Mining.

[13] Aurélien Géron. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.

[14] Nicola Guarino and Christopher Welty. 2000. A formal ontology of properties. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 97–112.

[15] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. 2017. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications* 73 (2017), 220–239.

[16] Steven Halim, Felix Halim, Steven S Skiena, and Miguel A Revilla. 2013. *Competitive Programming 3*. Lulu Independent Publish.

[17] Chowdhury Md Intisar and Yutaka Watanobe. 2018. Cluster analysis to estimate the difficulty of programming problems. In *Proceedings of the 3rd International Conference on Applications in Information Technology*. 23–28.

[18] Hermino Barbosa Freitas Junior, Filipe Dwan Pereira, Elaine Harada Teixeira Oliveira, David Fernandes, and Leandro Silva Galvão de Carvalho. 2020. Recomendação Automática de Problemas em Juízes Online Usando Processamento de Linguagem Natural e Análise Dirigida aos Dados. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*.

[19] Muhammad Khalifa and Noura Hussein. 2019. Ensemble Learning for Irony Detection in Arabic Tweets. In *Proceedings of the [email protected] In Metha P., Rosso P., Majumder P., Mitra M.(Eds.) Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2019). CEUR Workshop Proceedings.*

[20] Taisiya Kostareva, Svetlana Chuprina, and Alexandr Nam. 2016. Using Ontology-Driven Methods to Develop Frameworks for Tackling NLP Problems.. In *AIST (Supplement)*. 102–113.

[21] Carmen Lacave, Ana I Molina, and José A Cruz-Lemus. 2018. Learning Analytics to identify dropout factors of Computer Science studies through Bayesian networks. *Behaviour & Information Technology* 37, 10-11 (2018), 993–1007.

[22] Jarkko Lagus, Krista Longi, Arto Klami, and Arto Hellas. 2018. Transfer-learning methods in programming course outcome prediction. *ACM Transactions on Computing Education (TOCE)* 18, 4 (2018), 1–18.

[23] Paul Luo Li, Amy J Ko, and Andrew Begel. 2020. What distinguishes great software engineers? *Empirical Software Engineering* 25, 1 (2020), 322–352.

[24] Marcos Avner Pimenta de Lima, Leandro Silva Galvão de Carvalho, Elaine H. T. Oliveira, David Fernandes, and Filipe Dwan Pereira. 2020. Classificação de dificuldade de questões de programação com base em m´tricas de código. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, Vol. 31.

[25] Edward Ma. 2019. NLP Augmentation. https://github.com/makcedward/nlpaug.

[26] Harish Tayyar Madabushi, Elena Kochkina, and Michael Castelle. 2020. Cost-Sensitive BERT for Generalisable Sentence Classification with Imbalanced Data. *arXiv preprint arXiv:2003.11563* (2020).

[27] Raj P Mehta, Meet A Sanghvi, Darshin K Shah, and Artika Singh. 2020. Sentiment Analysis of Tweets Using Supervised Learning Algorithms. In *First International Conference on Sustainable Technologies for Computational Intelligence*. Springer, 323–338.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[29] Riichiro Mizoguchi and Jacqueline Bourdeau. 2000. Using ontological engineering to overcome common AI-ED problems. *Journal of Artificial Intelligence and Education* 11 (2000), 107–121.

[30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[31] Filipe Pereira, Elaine Oliveira, David Fernandes, Hermino Junior, and Leandro Silva Galvão Carvalho. 2019. Otimização e automação da predição precoce do desempenho de alunos que utilizam juízes online: uma abordagem com algoritmo genético. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, Vol. 30. 1451.

[32] Filipe Dwan Pereira, Samuel C Fonseca, Elaine HT Oliveira, David BF Oliveira, Alexandra I Cristea, and Leandro SG Carvalho. 2020. Deep learning for early performance prediction of introductory programming students: a comparative and explanatory study. *Brazilian journal of computers in education*. 28 (2020), 723–749.

[33] Filipe D Pereira, Elaine Oliveira, Alexandra Cristea, David Fernandes, Luciano Silva, Gene Aguiar, Ahmed Alamri, and Mohammad Alshehri. 2019. Early dropout prediction for programming courses supported by online judges. In *International Conference on Artificial Intelligence in Education*. Springer, 67–72.

[34] Filipe Dwan Pereira, Elaine HT Oliveira, David Fernandes, and Alexandra Cristea. 2019. Early performance prediction for CS1 course students using a combination of machine learning and an evolutionary algorithm. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, Vol. 2161. IEEE, 183–184.

[35] Filipe D Pereira, Elaine HT Oliveira, David BF Oliveira, Alexandra I Cristea, Leandro SG Carvalho, Samuel C Fonseca, Armando Toda, and Seiji Isotani. 2020. Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British Journal of Educational Technology* (2020).

[36] Keith Quille and Susan Bergin. 2019. CS1: how will they do? How can we help? A decade of research and practice. *Computer Science Education* 29, 2-3 (2019), 254–282.

[37] A. V. Robins. 2019. Novice programmers and introductory programming. In *The Cambridge Handbook of Computing Education Research*. Cambridge University Press, Cambridge, Chapter 12, 327–376.

[38] Alexander Joseph Romiszowski. 2016. *Designing instructional systems: Decision making in course planning and curriculum design*. Routledge.

[39] Antonio A Sánchez-Ruiz, Guillermo Jimenez-Diaz, Pedro P Gómez-Martín, and Marco A Gómez-Martín. 2017. Case-Based Recommendation for Online Judges Using Learning Itineraries. In *International Conference on Case-Based Reasoning*. Springer, 315–329.

[40] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. 2018. A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 1–34.

[41] Wayne Xin Zhao, Wenhui Zhang, Yulan He, Xing Xie, and Ji-Rong Wen. 2018. Automatically learning topics and difficulty levels of problems in online judge systems. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 1–33.